OverlayFS and its use in Yocto Project

Vyacheslav Yurkov

#ossummit

THE LINUX FOUNDATION

About me

- Use Linux since 2000.
- 15+ years of experience in different industries
- Regular contributor to open source projects (notably Linux Kernel, TF-A, Yocto Project, and others)
- Maintainer of OverlayFS classes in Openembedded Core

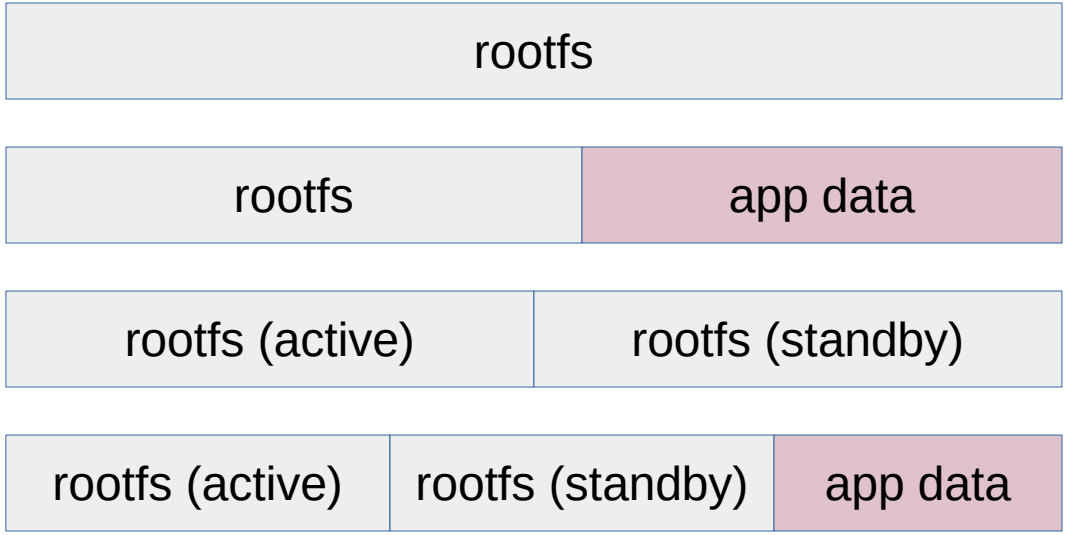
Vyacheslav Yurkov
(Slava)

Agenda

- Problem description
- What is OverlayFS
- Kernel requirements
- How it’s used in Yocto
- Debugging tools

Partition layout of embedded device

Possible layout

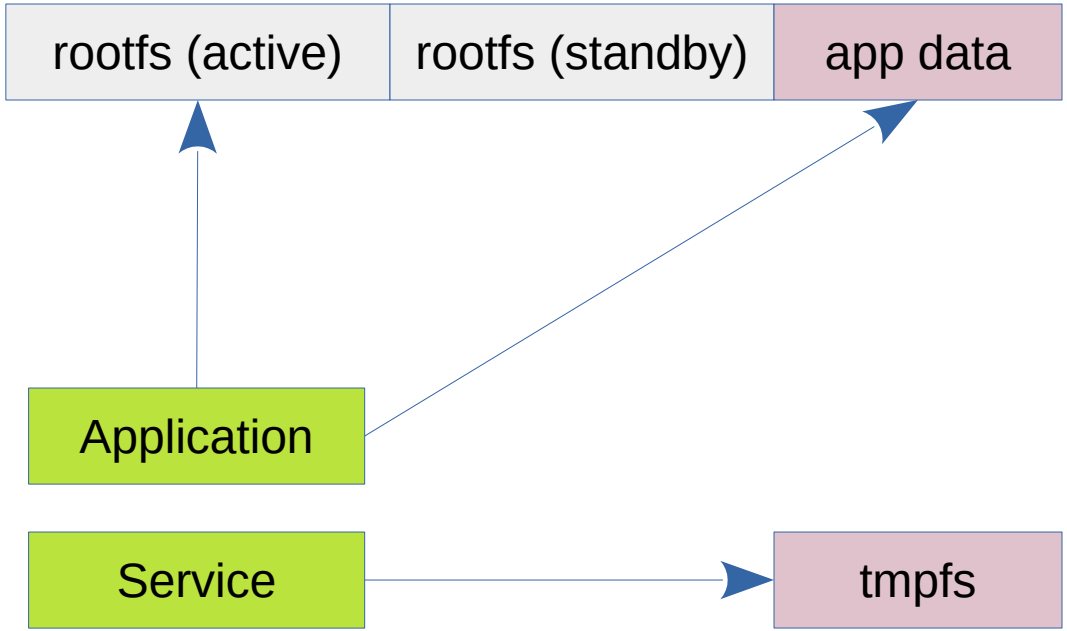


Update mechanism

- squashfs image(s)
- swupdate
- ostree

- swupdate
- rauc
- mender

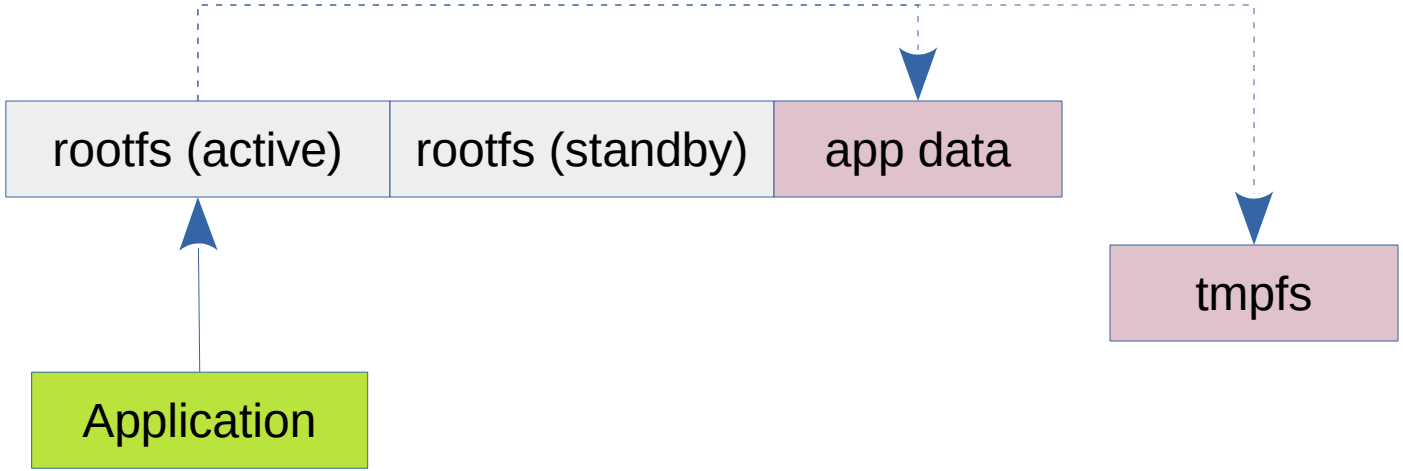
Application write access (part 1)



App / service writes:

1. To root fs (can be RO)
2. To <data> partition
3. To `tmpfs` mount point

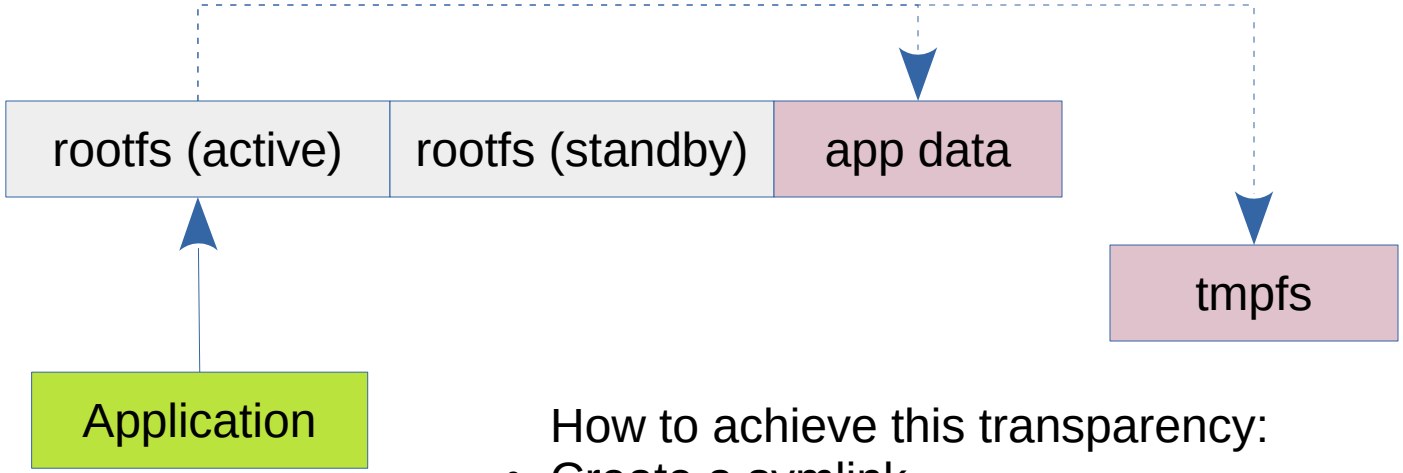
Application write access (part 2)



App writes to actual root fs:

1. You can't change the app
2. It requires a lot of time/effort to change it

Application write access (part 3)

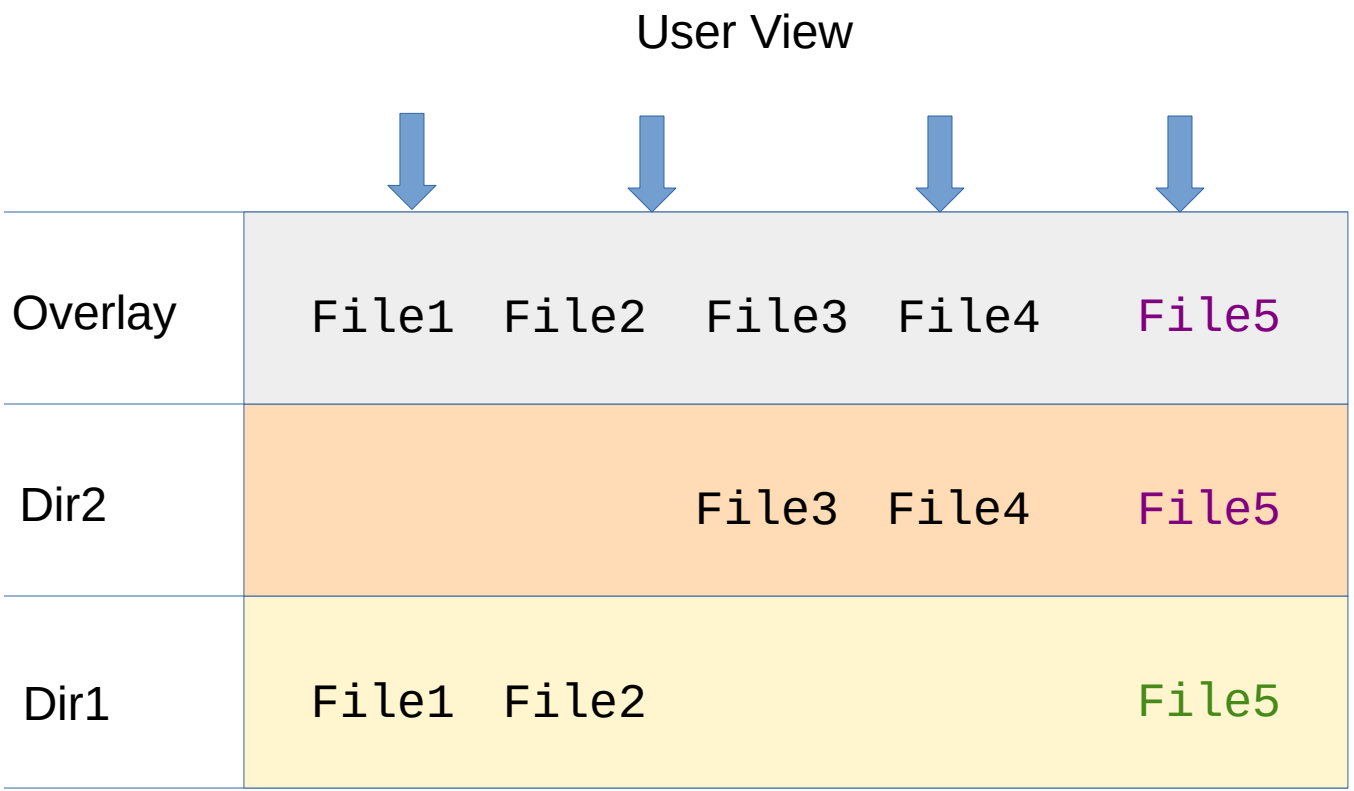


How to achieve this transparency:

- Create a symlink
- Bind mount required directory(s)
- Mount overlayfs for required directory(s)

What is OverlayFS

- OverlayFS is a union mount filesystem implementation for Linux
- Union mount is a combination of multiple directories into one that appears to contain their combined contents


User View
Overlay
File1 File2 File3 File4 File5
Dir2
File3 File4 File5
Dir1
File1 File2 File5

Terminology

- Lower layer / filesystem / directory tree - directory structure that appears “below” in the overlayfs implementation. It does not have to be writable. Can be another overlayfs.
- Upper layer / filesystem / diretory tree - directory structure that appears on-top of lower layer. Can be writable (in that case NFS is not supported)
- Merged / mounted overlay - overlayfs directory after mounting
- whiteouts / opaque objects - a file or directory marked as removed on the upper layer

OverlayFS example (part 1)

```
# ls -1 dir1
file1.txt
file2.txt
file5.txt

# ls -1 dir2
file3.txt
file4.txt
file5.txt

# cat dir1/file5.txt
file from dir1

# cat dir2/file5.txt
file from dir2

# mount -t overlay overlay -o
lowerdir=dir1,upperdir=dir2,workdir=work dir3
```

OverlayFS example (part 2)

```
# mount -t overlay overlay -o
lowerdir=dir1,upperdir=dir2,workdir=work dir3

# ls -1 dir3
file1.txt
file2.txt
file3.txt
file4.txt
file5.txt

# cat dir3/file5.txt
file from dir2
```

OverlayFS file removal

```
# rm -f dir3/file1.txt
# ls -1 dir3
file2.txt
file3.txt
file4.txt
file5.txt

# ls -1 dir1
file1.txt
file2.txt
file5.txt

# ls -l dir2
total 4
c--------- 2 root root  0, 0 Jul 29 19:55 file1.txt
-rw-r--r-- 1 uvv  users    0 Jul 27 21:21 file3.txt
-rw-r--r-- 1 uvv  users    0 Jul 27 21:21 file4.txt
-rw-r--r-- 1 root root    15 Jul 29 19:38 file5.txt
```

OverlayFS directory handling (part 1)

```
# umount dir3
# mkdir dir1/subdir1
# mkdir dir1/subdir2
# mkdir dir2/subdir2
# mount -t overlay overlay -o
lowerdir=dir1,upperdir=dir2,workdir=work dir3

# ls -1 dir3
file2.txt
file3.txt
file4.txt
file5.txt
subdir1/
subdir2/
```

OverlayFS directory handling (part 2)

```
# rmdir dir3/subdir1
# mkdir dir3/subdir1
# touch dir3/subdir2/file6.txt
# touch dir1/subdir2/file7.txt (offline!)

# getfattr -m- dir2/*
# file: dir2/subdir1
trusted.overlay.opaque

# file: dir2/subdir2
trusted.overlay.origin

# ls -1 dir3/subdir2
file6.txt
file7.txt
```

Kernel configuration (part 1)

For *defconfig* BSPs:

```
CONFIG_OVERLAY_FS=y
# CONFIG_OVERLAY_FS_REDIRECT_DIR is not set
# CONFIG_OVERLAY_FS_REDIRECT_ALWAYS_FOLLOW is not set
# CONFIG_OVERLAY_FS_INDEX is not set
# CONFIG_OVERLAY_FS_METACOPY is not set
```

Or if you use *linux-yocto* and/or BSP description files (`*.scc`):

```
KERNEL_EXTRA_FEATURES:append = "features/overlayfs/overlayfs.scc"
```

Kernel configuration (part 2)

Offline changes, when the overlay is not mounted, are allowed to the upper tree.

Offline changes to the lower tree are only allowed if the “metadata only copy up”, “inode index”, “xino” and “redirect_dir” features **have not been used**.

If the lower tree is modified and any of these features has been used, the behavior of the overlay is undefined, though it will not result in a crash or deadlock.

Requires kernel >= 5.15

How to use it in Yocto

- initramfs images
- volatile-binds.bb recipe
- overlayfs and overlayfs-etc bbclasses

Initramfs

- The recipe is called initramfs-framework (*meta/recipes-core/initrdscripts*)
- It contains a package initramfs-module-overlayroot
- This module requires `rootrw=<foo>` to be passed as a kernel parameter to specify the device/partition to be used as RW by the overlay
- It does not require the read-only IMAGE_FEATURE to be enabled.

volatile-binds (part 1)

Add *recipes-core/volatile-binds/volatile-binds.bbappend* to your layer:

```
VOLATILE_BINDS += "\
    /data/opt/myservive /opt/myservice\n\
"
```

volatile-binds (part 2)

- Already included as run-time dependency of *systemd*
- Can enable/disable mount points on a ***layer*** basis
- Tries to mount *overlayfs*, if not successful then rolls back to *bind* mounts
- Works only if lower layer FS is read-only
- You have to know it exists (not described in the docs)

overlayfs.bbclass (part 1)

application.bb recipe:

```
inherit overlayfs
OVERLAYFS_WRITABLE_PATHS[data] += "/opt/myservice"
```

Distro configuration:

```
DISTRO_FEATURES:append = " overlayfs"
```

Machine configuration:

```
OVERLAYFS_MOUNT_POINT[data] = "/data"
```

overlayfs.bbclass (part 2)

- Can enable/disable mount points on a ***recipe*** basis
- Works regardless whether lower layer FS is RO or RW
- QA checks confirms you have systemd mount unit for /data partition defined elsewhere in your BSP (e.g. in systemd-machine-units recipe) and it’s installed into the image (can be skipped if required)
- Documented and unit tested
- Works for *systemd* only

overlayfs-etc.bbclass

image.bb recipe:

```
EXTRA_IMAGE_FEATURES += "overlayfs-etc"
```

Machine configuration:

```
OVERLAYFS_ETC_MOUNT_POINT = "/data"
OVERLAYFS_ETC_DEVICE = "/dev/mmcblk0p1"
```

Useful tools / debugging techniques (part 1)

https://github.com/hisilicon/overlayfs-progs

- Provides *fsck.overlay* utility
- Recipe available in meta-openembedded/meta-filesystems/recipes-utils/overlayfs

Useful tools / debugging techniques (part 2)

https://github.com/kmxz/overlayfs-tools

- Provides utilities:
  - **`vacuum`** - remove duplicated files in upperdir
  - **`diff`** - show the list of modified files (between `overlay` and `lowerdir`)
  - **`merge`** - merge down the changes from `upperdir` to `lowerdir`
  - **`deref`** - copy changes from `upperdir` to `uppernew`
- Recipe available in meta-openembedded/meta-filesystems/recipes-utils/overlayfs

Makes sense to merge both projects?

Useful tools / debugging techniques (part 3)

https://git.kernel.org/pub/scm/fs/xfs/xfstests-dev.git/tree/

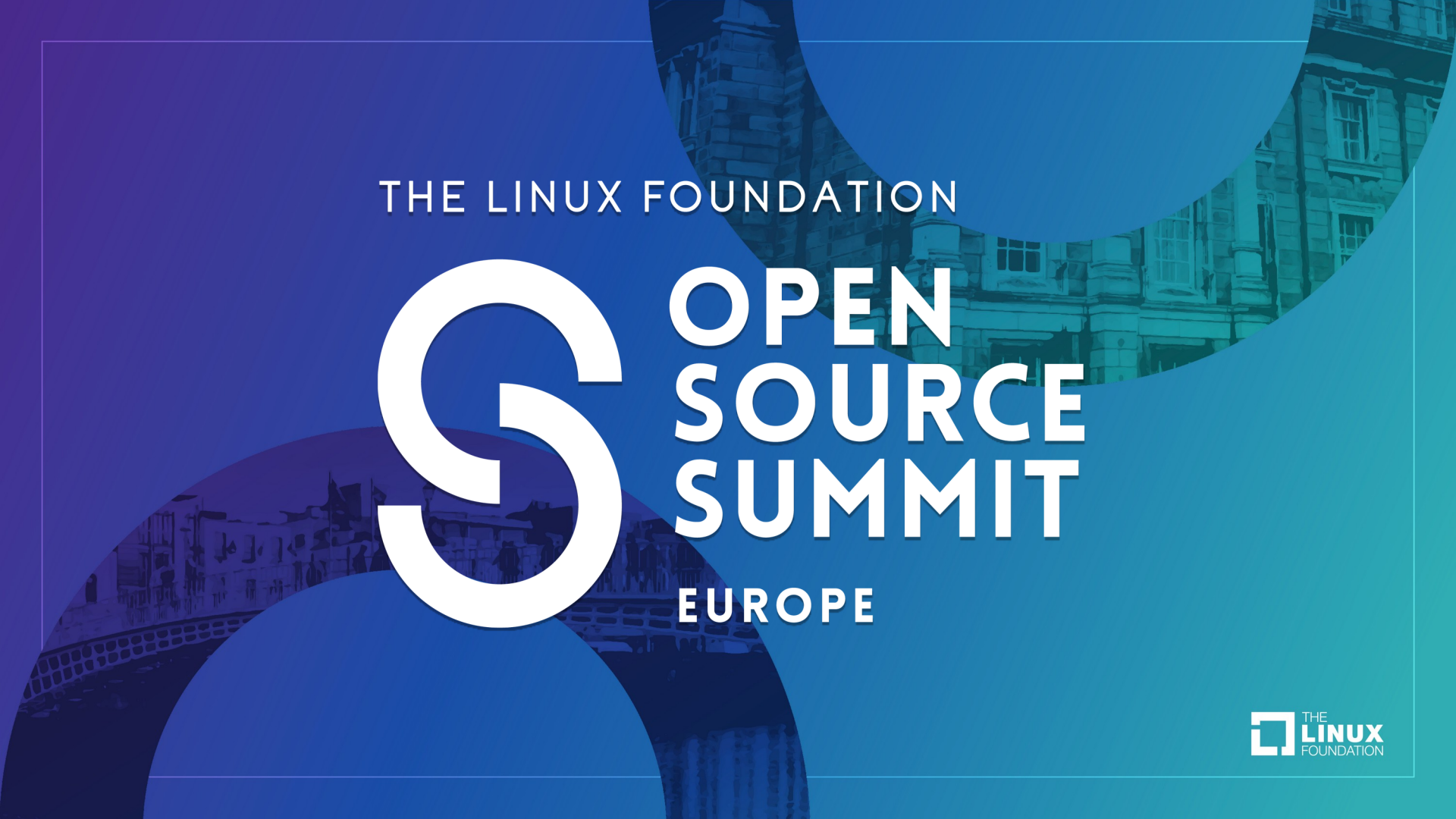
- Provides a file system QA test suite
- Recipe available in meta-openembedded/meta-filesystems/recipes-utils/xfstests, sets up required test environment
- OverlayFS patches / fixes should be sent to linux-unionfs@vger.kernel.org

References

- https://lwn.net/Articles/312641/
- https://www.kernel.org/doc/html/latest/filesystems/overlayfs.html

Thanks to

- Miklos Szeredi
- Amir Goldstein


THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
EUROPE
THE LINUX FOUNDATION